METHOD AND SYSTEM FOR PROVIDING REMOTE PROCEDURE CALLS IN A MULTIPROCESSING SYSTEM

FIELD OF THE INVENTION

The present invention relates to multiprocessing systems and more particularly to a method and system for providing remote procedure calls in such systems.

BACKGROUND OF THE INVENTION

The multiprocessor, parallel execution model used in a general network processor design demands an effective, simple implementation of remote procedure calls. Remote calls may be made from a general purpose processor to a network processor pico-engine, or generically, between any processors within the system. The execution of remote procedure calls enables flexible optimized distribution of both function and workload among multiple processors. Two methods are currently utilized to provide this optimized distribution.

One solution is to pass the entire instruction address within the control message. This successfully effects the remote procedure call, but how does the originator of the message know the instruction address of the call? If the processors use independent code loads, what if the entry point for the remote procedure call execution changes from code load to code load? Some external processing is required to analyze code images to verify/determine entry points and assist in resolving entry point to address at remote procedure call issue time.

An alternative solution is to create a jump table for the processor executing the remote procedure call. A remote procedure call is requested by issuing a request with the appropriate index into the jump table. Thus, the only consistency required between remote procedure call network processor requester and remote procedure call network processor receiver is that the

table index remain consistent in both processes. This approach works well, but suffers from requiring a pre-allocated jump table. In the multiprocessor environment memory storage is at a premium. Thus it is desirable to minimize the amount of dedicated storage in the memory.

Accordingly, what is needed is a system and method which overcomes the above-identified problem. The present invention addresses such a need.

SUMMARY OF THE INVENTION

A method and system for providing remote procedure calls in a multiprocessing system is disclosed. The multiprocessing system includes a general purpose processor and a plurality of network processors. Each of the plurality of network processors includes a memory. The method and system comprises accessing a reserved address in at least one of the network processors and initiating a software action by a first portion of the reserved address. The method and system further includes pointing to an address within the memory of the at least one network processor to be processed based upon data in a second portion of the reserved address, wherein the data at the address is processed.

A system and method in accordance with the present invention provides an indirect software jump in a microprocessing system through providing a reserved address in memory of each of the reserved address includes two portions. A first portion of the address triggers the software event requested by the general purpose processor (for example) and a second portion of the reserved address is utilized to process the data that was loaded at that address in the processor. The indirect software jump allows a general purpose processor to execute software on a network processor indirectly for custom application services or debug operations.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a conventional multiprocessing system.

Figure 2 illustrates such a memory architecture.

Figure 3 illustrates the use of a jump table resource on the network processor

Figure 4 is a block diagram illustrating a jump table.

Figure 5 is a diagram of a reserved address within a network processor utilized to provide the indirect software jump.

Figure 6 is a flow chart for providing an indirect software jump in a multi-processing system.

DETAILED DESCRIPTION

The present invention relates to multiprocessing systems and more particularly to a method and system for providing remote procedure calls in such systems. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Figure 1 illustrates a conventional multiprocessing system 10. The conventional multiprocessing system 10 includes a general purpose processor 12 which includes an instruction set. The general purpose processor 12 controls a plurality of network processors 14, 16 and 18 via a switch fabric 20. Each of the network processors 14 through 18 are

independent central processing units. The network processors 14, 16, 18 are special processors that provide protocol forwarding and are highly tuned for network programming. The switch fabric 20 allows for combinations of processors 12-18 to communicate with each other.

In each of the processors 12-18, there is dedicated memory which holds data and code. Figure 2 illustrates such a memory architecture 100. As is seen data 102 comprises upper portion of the memory structure and code 104 for executing the data occupies the lower portion. The network processors 14-18 (Figure 1) could be the same version of a processor (such x86, Apple, Pentium or RISC or other processor architecture). However, it is more likely that the network processors are different versions of a particular processor family. Accordingly, the memory architecture may be different for each processor so a software operation that is called by one processor to be preformed b other processors may have to be initiated from a different part of the memory architecture from processor to processor. In addition, memory is a very precious resource, particularly for the network processors. The network processors are highly tuned for protocol handling and therefore do not have a large amount of memory to dedicate to a particular function. As the number of network processors grow, the number of commands that can be sent to a processor increases.

The multiprocessor, parallel execution model used in a network processor design demands an effective, simple implementation of remote procedure calls. Remote calls may be made from the general purpose processor 12 to one or more network processor 14-18, or generically, between any of the processors 12-18 within the system 10. The execution of remote procedure calls enables flexible optimized distribution of both function and workload among multiple processors. To provide for this flexibility, one of two conventional systems are utilized.

In a first system, the entire instruction address is passed within a control message. This successfully provides for the remote procedure call, but the originator of the message does not know the instruction address of the call. When the processors use different independent code loads, sometimes the entry point for the remote procedure call execution changes from code load on one processor to a different code load on a second processor. Some external processing is required to analyze code images to verify/determine entry points and assist in resolving entry point to address at remote procedure call issue time for each of the processor.

An alternative system is to create a jump table for the processor executing the remote procedure call. Figure 3 illustrates the use of a jump table resource on a network processor 14'. A remote procedure call is requested by issuing a request with the appropriate index into the jump table 200 via the general purpose processor 12'. Thus, the only consistency required between the remote procedure call requester and remote procedure call processor is that the table index remain consistent in both processors. This approach works well, but suffers from requiring a pre-allocated jump table 200.

To explain the problem with a pre-allocated jump table 200, refer now to the following discussion in conjunction with the accompanying figures. Figure 4 is a block diagram illustrating a jump table 200. The jump table 200 includes a plurality of entries which each of which includes an offset and an address. The jump table 200 must be accessed by a control message from a processor that indicates a software action. Next the processor must load a base address and add that to the starting address. Thereafter, the software executes from this new address. Accordingly contiguous memory blocks must be allocated in the memory for the jump table and there is always some of the table that is not used and therefore can be utilized for no other purpose.

Accordingly, a significant amount of reserved storage for either maintaining a pre-allocated jump table or a large amount of external processing is required to analyze code images to determine what the starting instruction address is for the software action adding to the latecy of the system.

In a system and method in accordance with the present invention an indirect software jump is provided through reserving an address in memory where the address includes two portions. A first portion triggers the software event and a second portion of the address is utilized to process the data that was loaded into a register in the processor. An indirect software jump allows a general purpose processor to execute software on a network processor indirectly for custom application services or debug. An example of such a service is performing a custom table insert which inserts a forwarding entry and automatically inserts a pointer to a global counter block specific to that network processor.

To describe the present invention in more detail, refer now to the following discussion in conjunction with the accompanying figures. Figure 5 is a diagram of a reserved address 300 within a network processor utilized to provide the indirect software jump. The reserved address 300 is typically a four to eight byte value dependent upon the length of the instruction. The address includes a first portion 302 for triggering the software action and a second portion 304 for pointing to the address of memory that is to be processed. Each of the reserved addresses 300 within a processor are located in a known area (preferably in the same location for each memory).

Figure 6 is a flow chart for providing an indirect software jump in a multi-processing system. First, a reserved address is accessed by a command, via step 402. Next, a software action is initiated by a first portion of the reserved address, via step 404. Then, the second

portion of the address points to an address that is to be processed within the memory, and wherein the data at that address is processed, via step 406.

By utilizing an indirect software jump, the remote procedure call executor can de-reference the requested operation within the scope of its own load module. External processing is not required to resolve entry points. A preallocated jump table is not precluded, but is not required. Since the full address of the remote procedure call (or function) pointer is provided in the place of an index, the pointers can be arranged in a tabular form or anchored individually throughout the memory space. The only requirement is that the address of the function pointer, or remote procedure call pointer, remain constant between remote procedure call caller and processor. Therefore, allocation of memory is minimized and contiguous memory to execute an indirect software jump is also not required.

Conclusion

A system and method in accordance with the present invention provides an indirect software jump in a microprocessing system through providing a reserved address in memory of each of the reserved address includes two portions. A first portion of the address triggers the software event requested by the general purpose processor (for example) and a second portion of the reserved address is utilized to process the data that was loaded at that address in the processor. The indirect software jump allows a general purpose processor to execute software on a network processor indirectly for custom application services or debug operations.

Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be

variations to the embodiments and those variations would be within the spirit and scope of the present invention. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.